**Name: Mayank Pandey** Roll No -27

**Class: MSC CS Part-1 Subject: Bioinformatics**

**Academic Year: 2020-2021**

|  |  |  |  |
| --- | --- | --- | --- |
| **Sr. No** | **Practical** | **Page No** | **Signature** |
| **1.** | **Pairwise Alignment** | **2** |  |
| **2.** | **Find Identity of Sequence** | **4** |  |
| **3.** | **Similarity of Sequence** | **6** |  |
| **4.** | **Percentage matching of sequence** | **8** |  |
| **5.** | **find motif in a given sequence** | **10** |  |
| **6.** | **Perform BLAST search and Find the no of repetition of each nucleotide in the sequence.** | **12** |  |
| **7.** | **Enter six protein sequence of different organism and write a program**  **to find a fingerprint of sequence.** | **19** |  |
| **8.** | **Find Regular Expression of given sequences.** | **21** |  |

**Practical 1- Pairwise Alignment**

**Source Code:**

se1 = input("Enter the first sequence: ")

se2 = input("Enter the second sequence: ")

seq1 = list(se1)

seq2 = list(se2)

score =[]

def Pairwise\_alignment(a,b):

gap(a,b)

print(a)

print(b)

value =0

length = len(a)

for i in range(0, length):

if (a[i] == b[i]):

score.append('1')

value =value+1

else:

score.append('0')

print(score)

print(value)

def gap(a,b):

if (len(a)==len(b)):

print()

else:

k= int(input("Enter the position to insert: "))

if(len(a)<len(b)):

a.insert(k,'-')

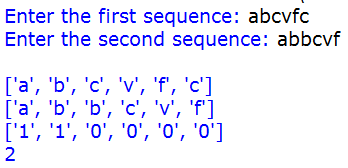
else:

b.insert(k,'-')

return (a,b)

Pairwise\_alignment(seq1,seq2)

**Output:**



**Practical 2- Find Identity of Sequence**

**Source Code:**

se1 = input("Ente rthe first sequence: ")

se2 = input("Ente rthe second sequence: ")

seq1 = list(se1)

seq2 = list(se2)

def find\_identity(a,b):

gap(a,b)

print(a)

print(b)

score= 0

length=len(a)

total\_elements = len(a)\*len(b)

for i in range(0,length):

for j in range(0,length):

if(a[i]==b[j]):

score=score+1

identity = (score/total\_elements)\*100

print("Matching Score: ", score)

print("Identity of the sequences: ",identity)

def gap(a,b):

if(len(a)==len(b)):

print()

else:

k=int(input("Enter the position to insert gap: "))

if(len(a)<len(b)):

a.insert(k,'-')

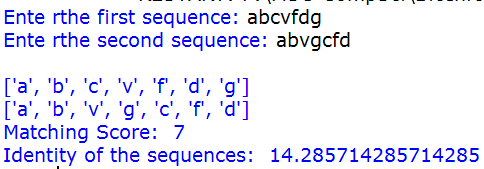
else:

b.insert(k,'-')

return (a,b)

find\_identity(seq1,seq2)

**Output:**



**Practical 3- Similarity of Sequence**

**Source code:**

se1 = input("Ente rthe first sequence: ")

se2 = input("Ente rthe second sequence: ")

how\_many = int(input("How many elements for similarity conditions?"))

similarities = []

for i in range(0,how\_many):

a=input("Enter an element: ")

c=int(input("How many elements is it similar to? "))

similarities.append([])

similarities[i].append(a)

for j in range(0,c):

b=input("What is it similar to? ")

similarities[i].append(b)

def compare(o,t,s):

print(o)

print(t)

print(s)

#checking if similar

score= 0

for i in range(len(o)):

for j in range(len(s)):

if o[i] in s[j] and t[i] in s[j] and o[i] != t[i]:

score += 1

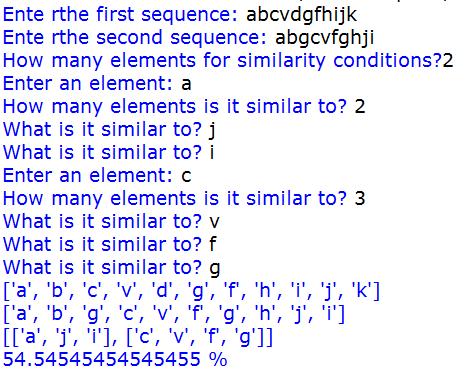
#calculating similarity

similarity = (score\*100)/len(o)

return similarity

print(compare(list(se1),list(se2),similarities),"%")

**Output:**



**Practical 4- Percentage matching of sequence**

**Source code:**

seq1=list(input("Enter the First sequence::"))

seq2=list(input("Enter the Second sequence::"))

#for identity

score=[]

def identity(a,b):

value=0

length=len(a)

for i in range(0,length):

if(a[i]==b[i]):

score.append('1')

value=value+1

else:

score.append('0')

return value

#for similarity

how\_many=int(input("How many elements for similarity condition?"))

similarities=[]

for i in range(0,how\_many):

a=input("Enter an element: ")

c=int(input("How many elements is it similar to? "))

similarities.append([])

similarities[i].append(a)

for j in range(0,c):

b=input("What is it similar to? ")

similarities[i].append(b)

def compare(o,t,s):

print(o)

print(t)

print(s)

#checking if similar

score=0

for i in range(len(o)):

for j in range(len(s)):

if o[i] in s[j] and t[i] in s[j] and o[i] != t[i]:

score+=1

#calculating similarity

similarity= score

return similarity

identi=(identity((seq1),(seq2)))

simi=(compare((seq1),(seq2),similarities))

print("similarity of the Sequences::",simi)

print("identity of the Sequences::",identi)

total=identi + simi

#counting

gap\_in\_sequence1=seq1.count('-')

gap\_in\_sequence2=seq2.count('-')

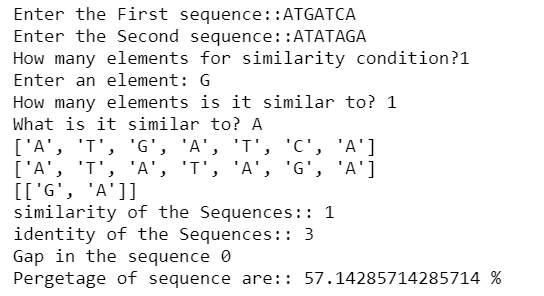
total\_gap=gap\_in\_sequence1 + gap\_in\_sequence2

print("Gap in the sequence",total\_gap)

pergetage\_of\_sequence=(total/(len(seq1)- total\_gap))\*100

print("Pergetage of sequence are::",pergetage\_of\_sequence,"%")

**Output:**



**Practical 5- Find motif in a given sequence**

**Source code:**

import random

l=int(input("Enter the length of motif"))

file=open("mot.txt","r")

r=file.read()

print("Sequence",r)

size=len(r)

print("Size of the sequence",size)

pos=random.randint(0,len(r)-5)

#pos=1

print("Position",pos)

motif=r[pos:pos+l]

print("Motif",motif)

i=pos+1

while(i<=size-1):

if(motif==r[i:i+1]):

str1=r[i:i+1]

print("Match motif",str1)

file1=open("motoutput.txt","a")

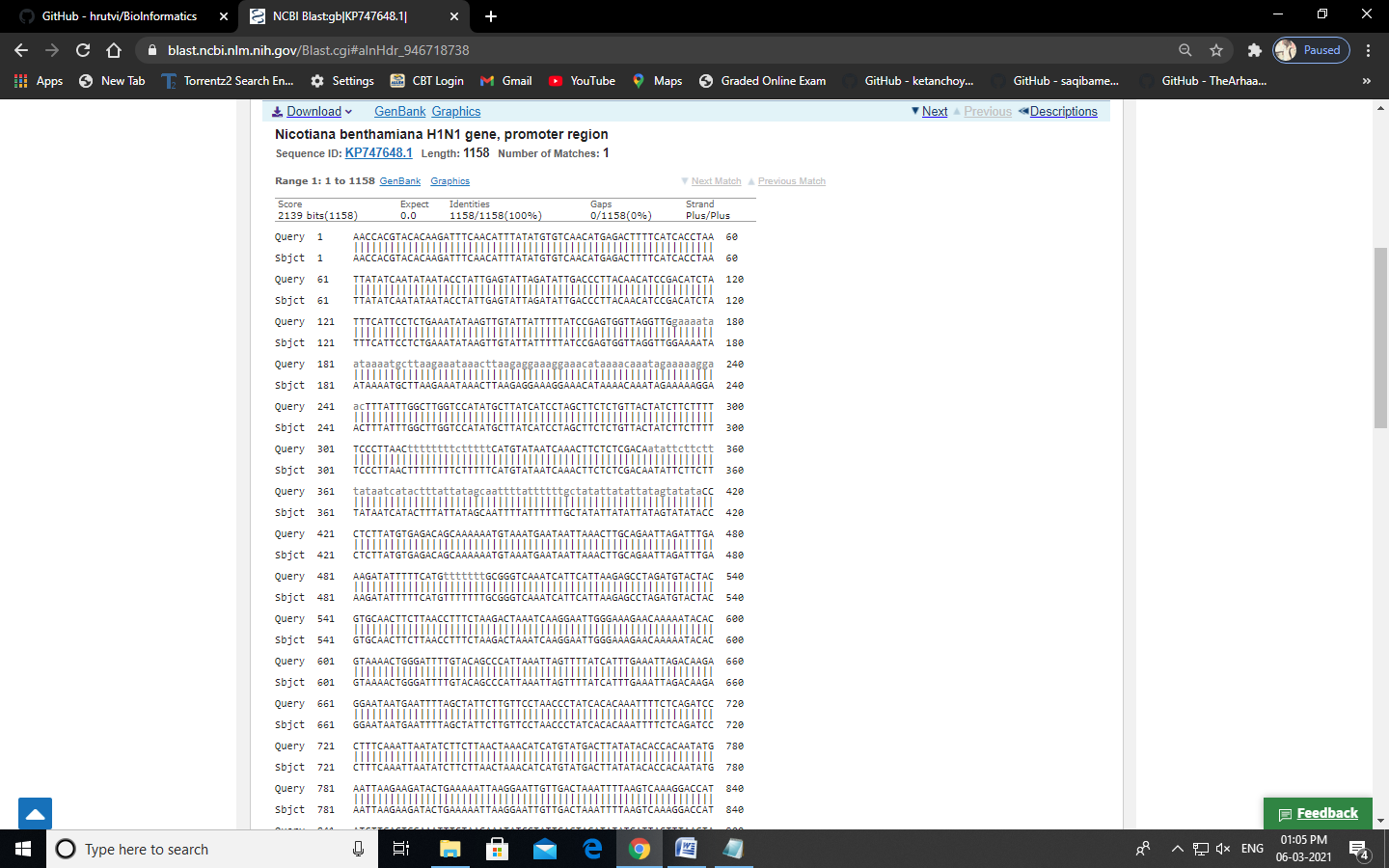
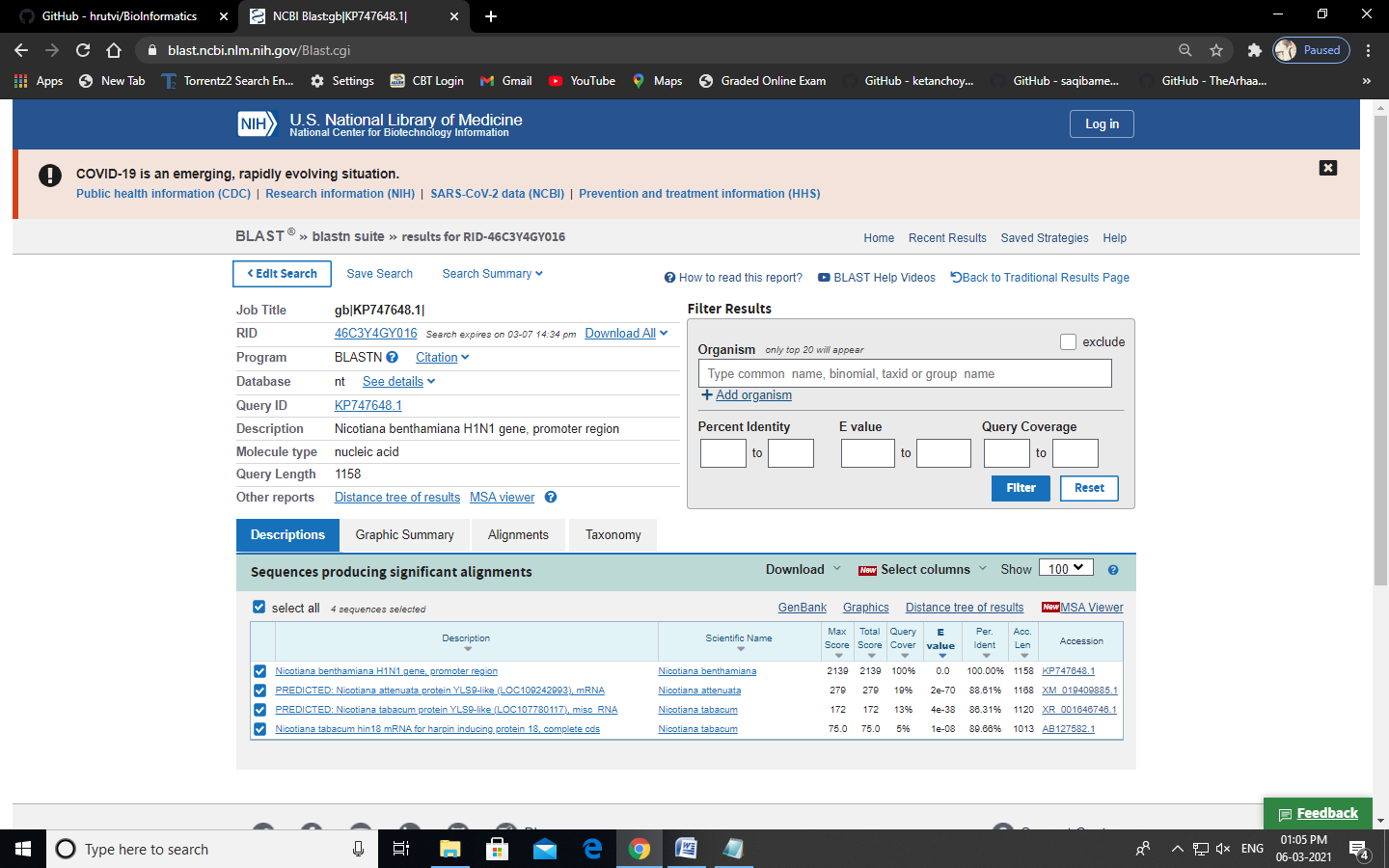
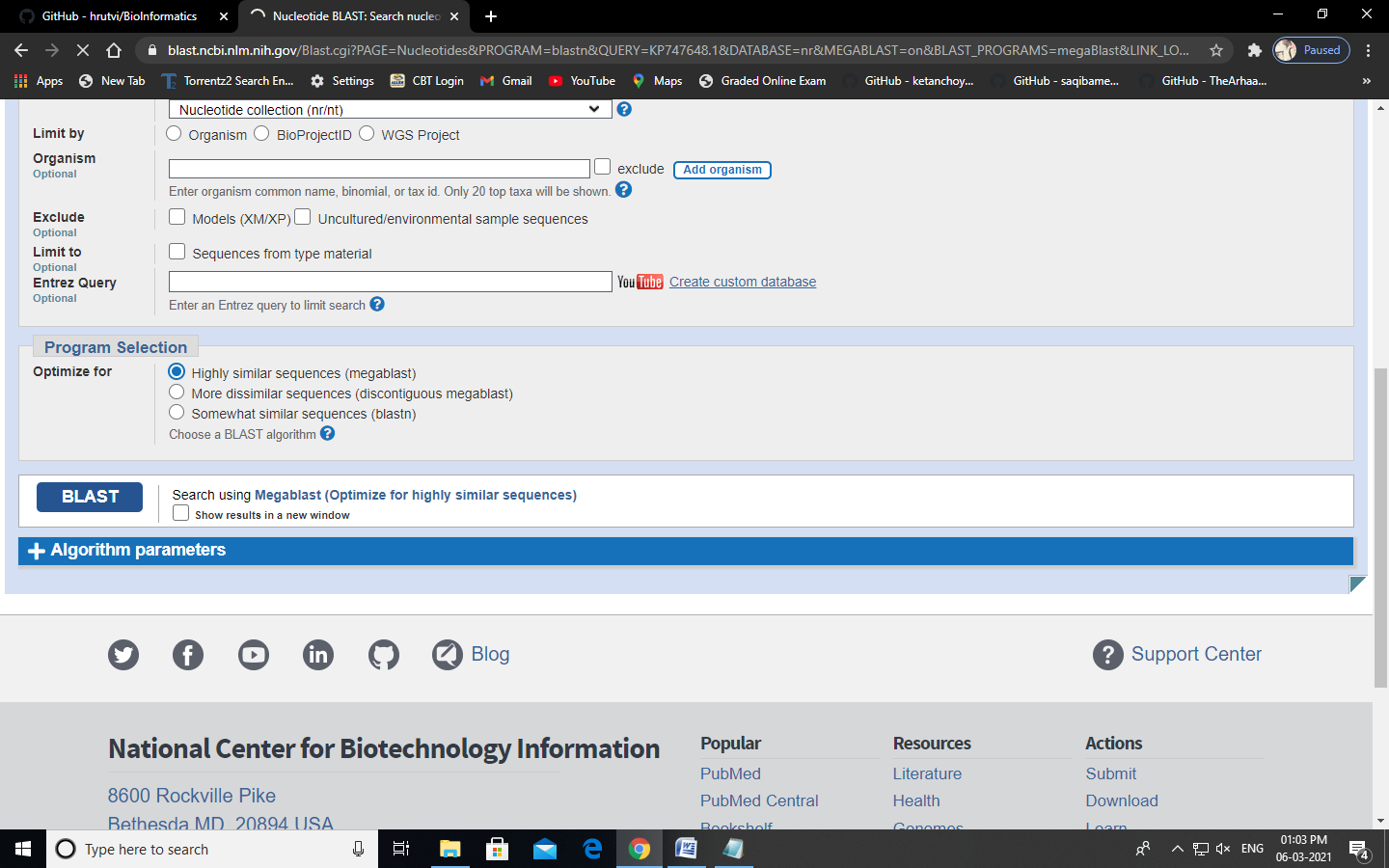
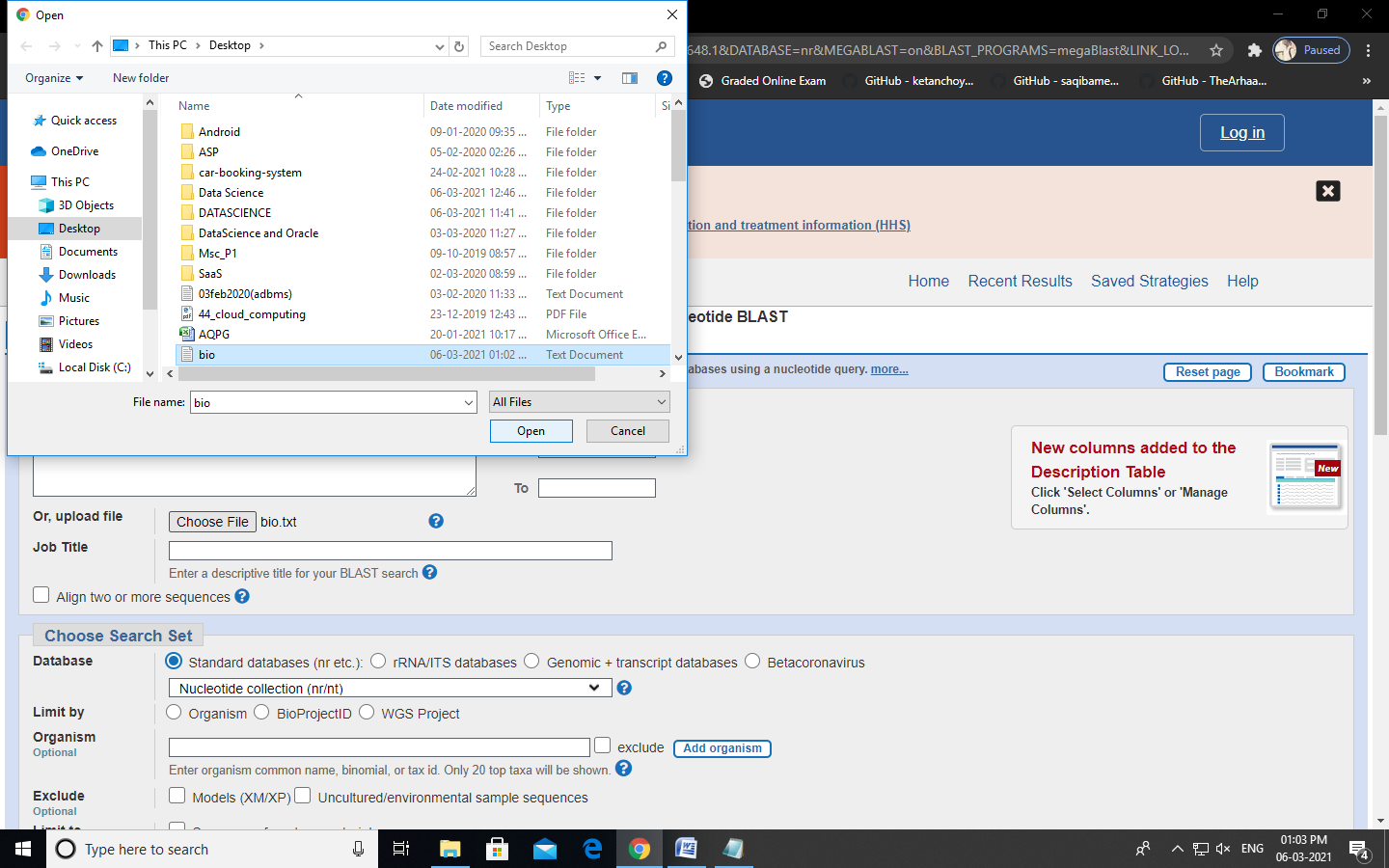
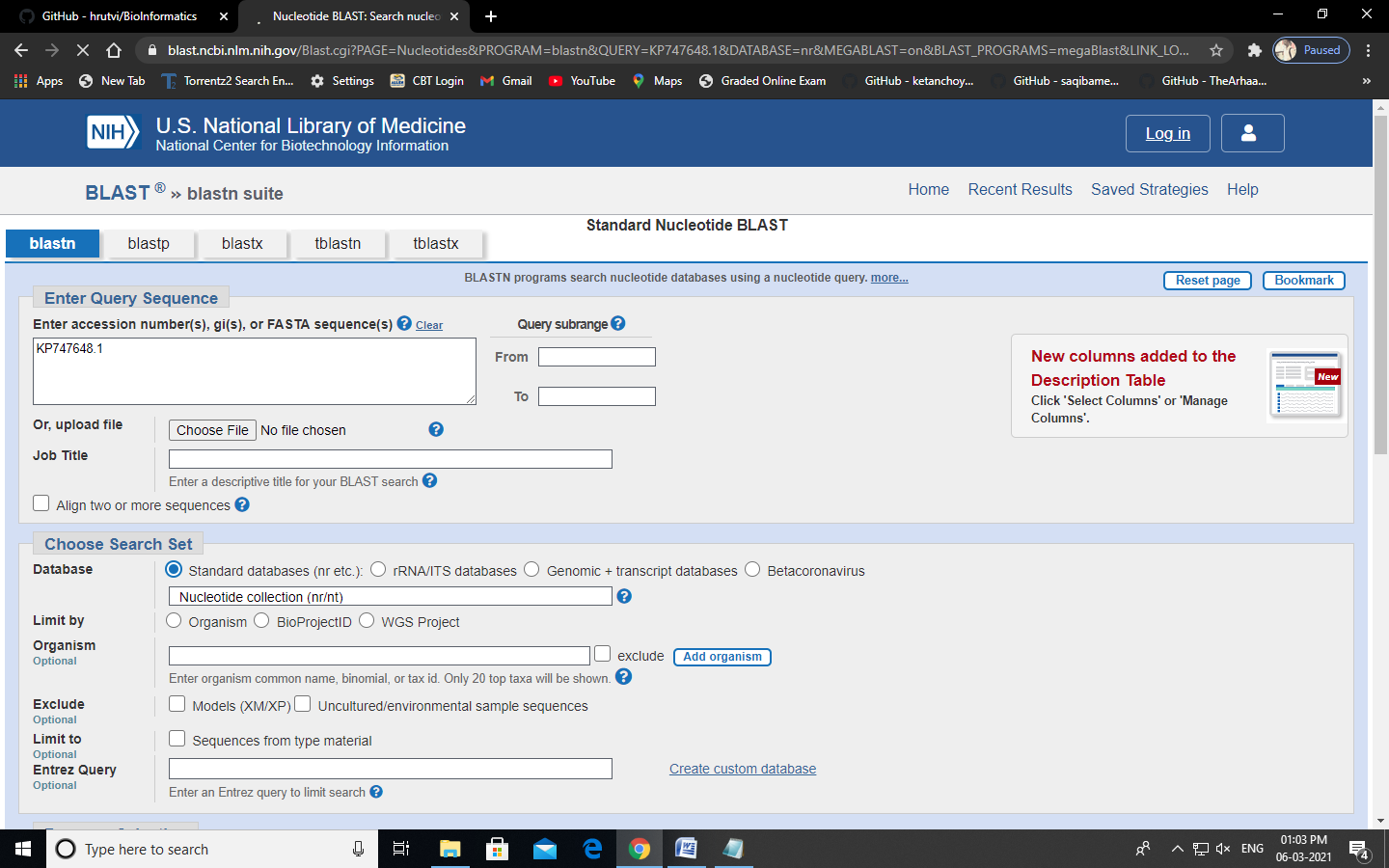
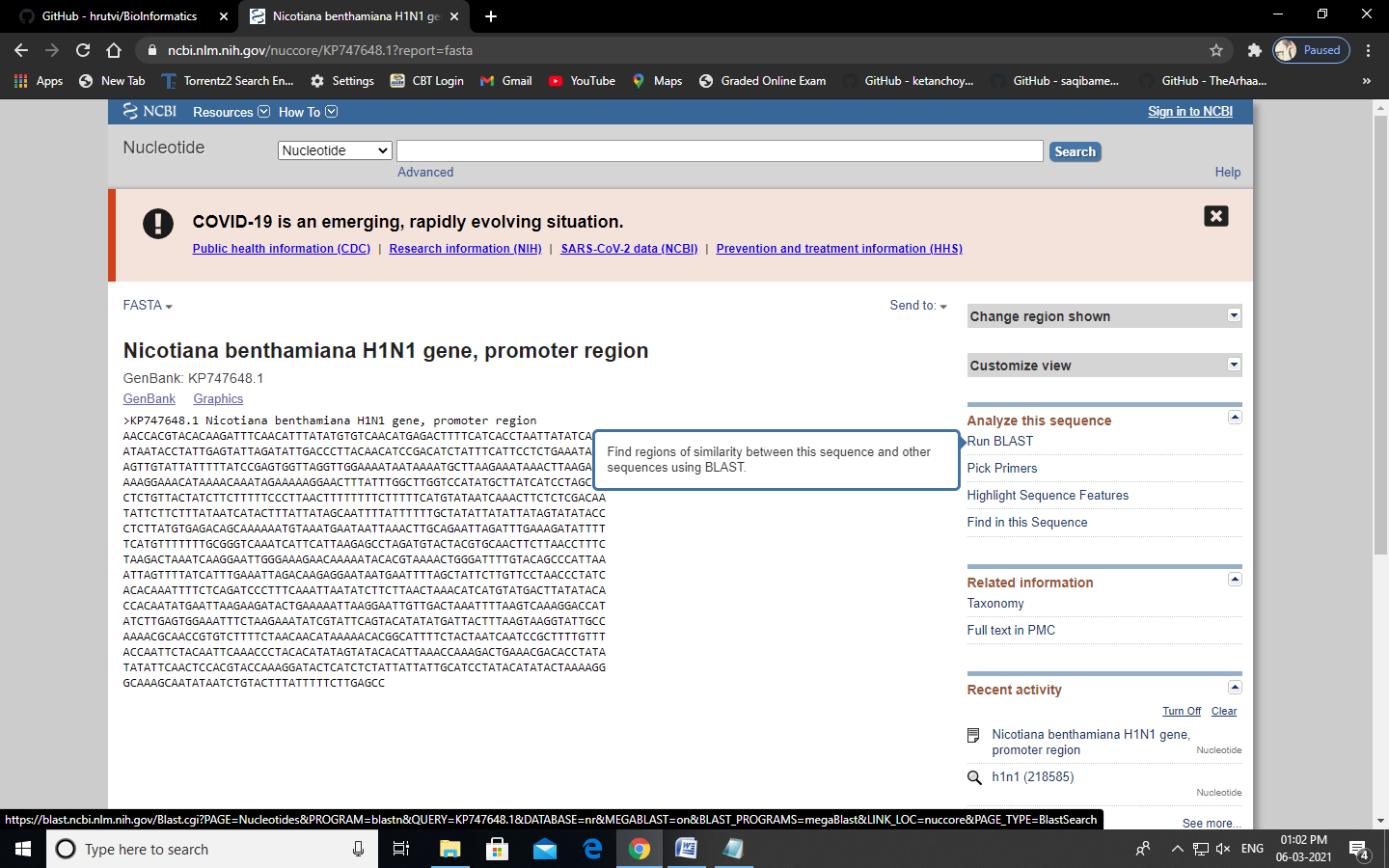
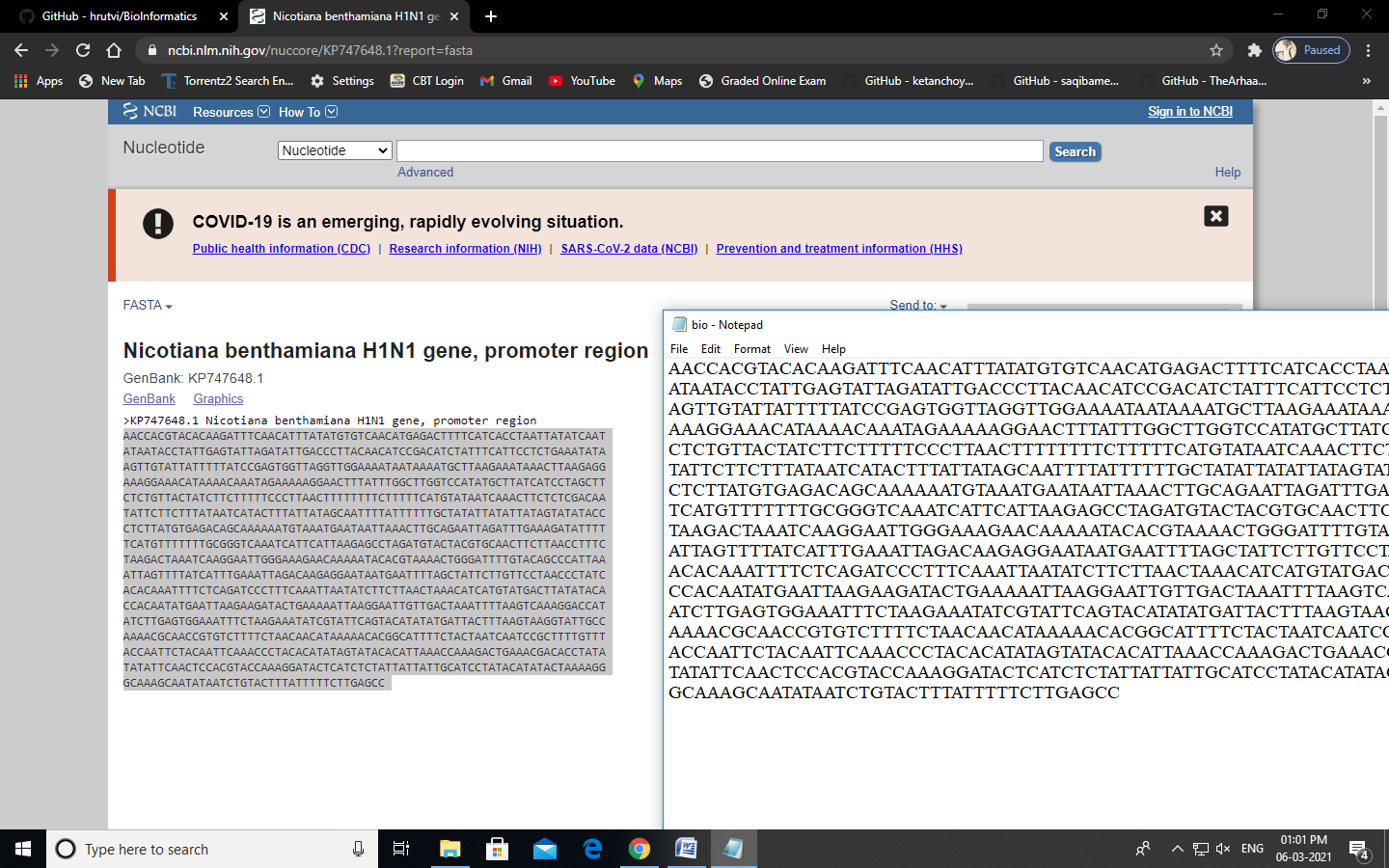
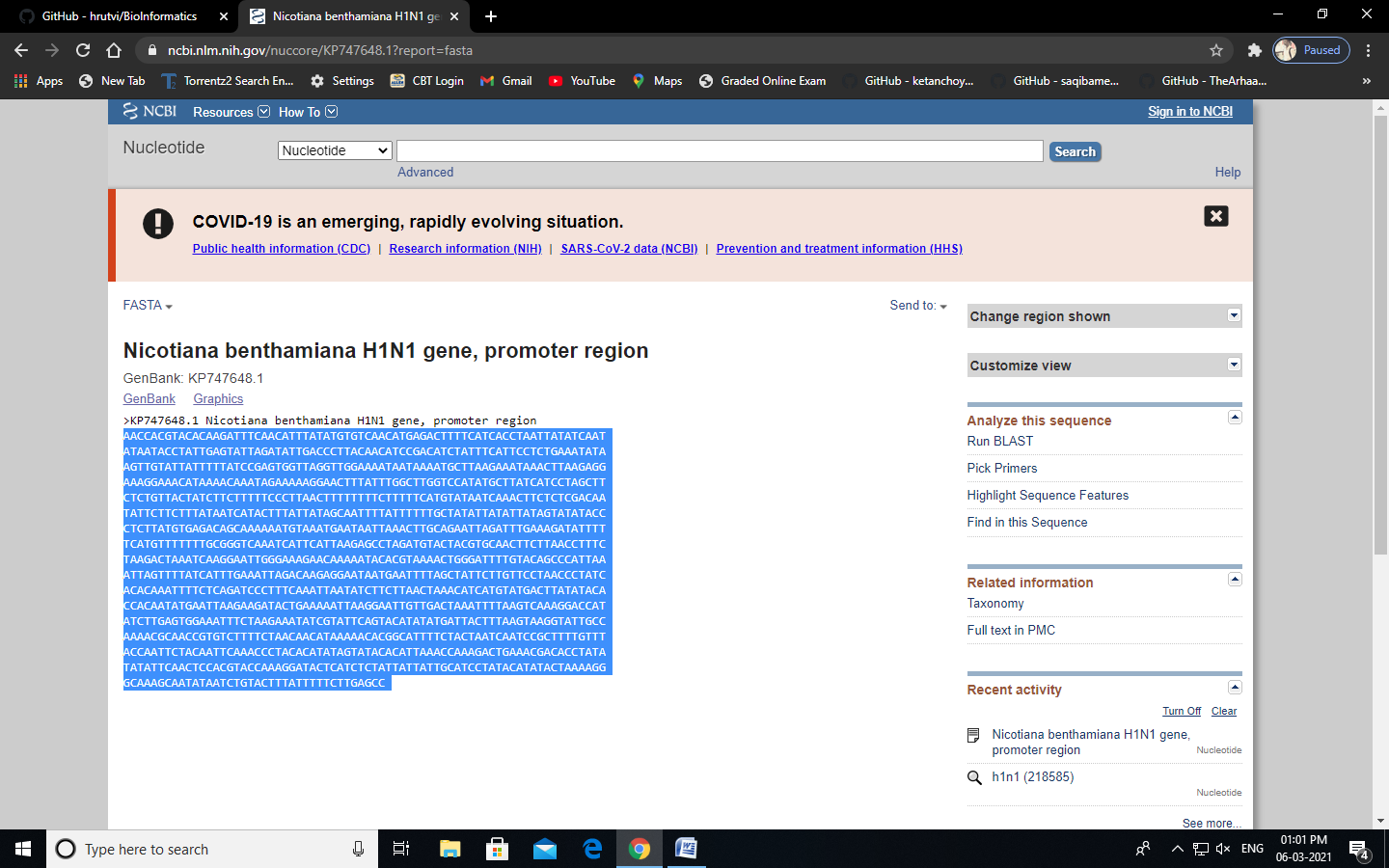
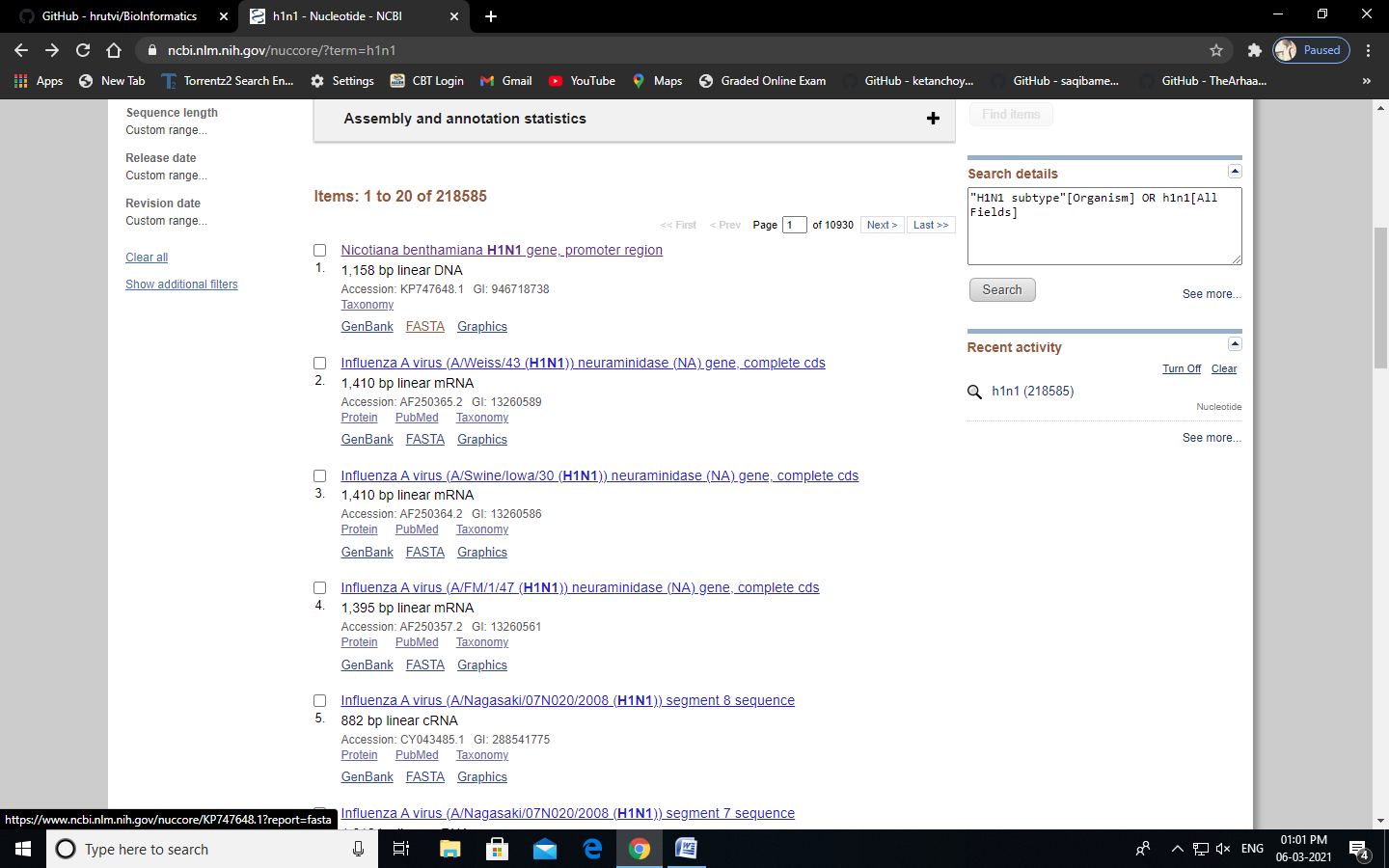
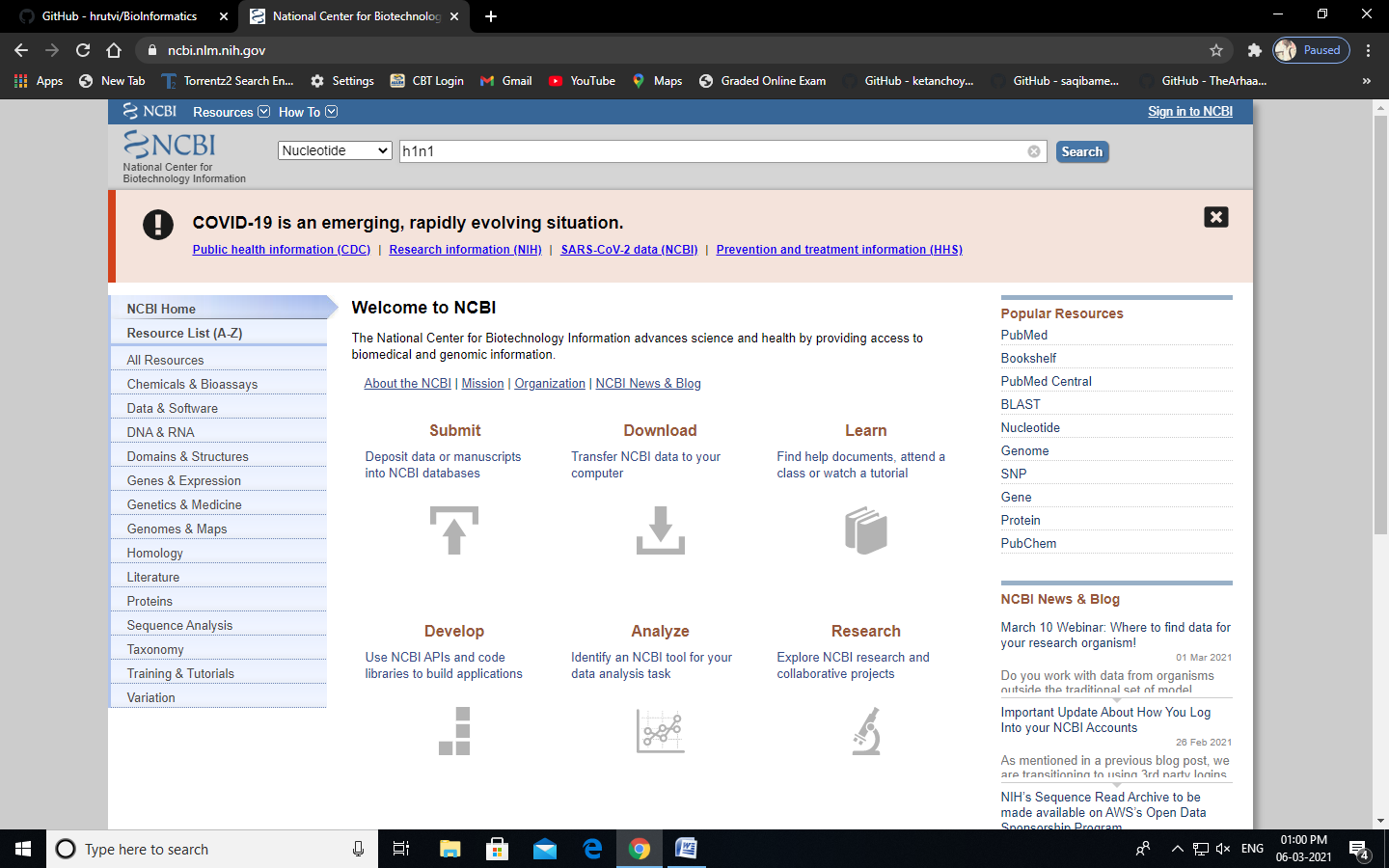
file1.write(str1+" ")

i+=1

**Output:**



**Practical 6 - Perform BLAST search and Find the no of repetition of each nucleotide in the sequence.**



**Source code:**

file=open("genes.txt","r")

r=file.read()

size=len(r)

score\_A=0

score\_C=0

score\_T=0

score\_G=0

for i in range(size):

if(r[i]=='A'):

score\_A+=1

elif (r[i]=='C'):

score\_C+=1

elif (r[i]=='T'):

score\_T+=1

elif (r[i]=='G'):

score\_G+=1

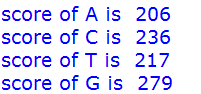
print("score of A is ",score\_A)

print("score of C is ",score\_C)

print("score of T is ",score\_T)

print("score of G is ",score\_G)

**Output:**



**Practical 7 - Enter six protein sequence of different organism and write a program**

**to find a fingerprint of sequence.**

**Source code:**

def solve\_fingerprint(seq\_list, no\_of\_col):

seq\_dict=dict()

for colnum in range(no\_of\_col):

counta,countc,countt,countg=0,0,0,0

for colseq in seq\_list:

if colseq[colnum]=='A':

counta+=1

elif colseq[colnum]=='T':

countt+=1

elif colseq[colnum]=='C':

countc+=1

elif colseq[colnum]=='G':

countg+=1

seq\_dict[colnum]=[counta,countc,countt,countg]

display\_results(seq\_dict)

def display\_results(seq\_dict):

print("\tA \tC \tT \tG")

for key in seq\_dict:

print("\n",\*seq\_dict[key],sep="\t")

no\_of\_seq=int(input("Enter the number of sequence: "))

print("Enter all the sequences")

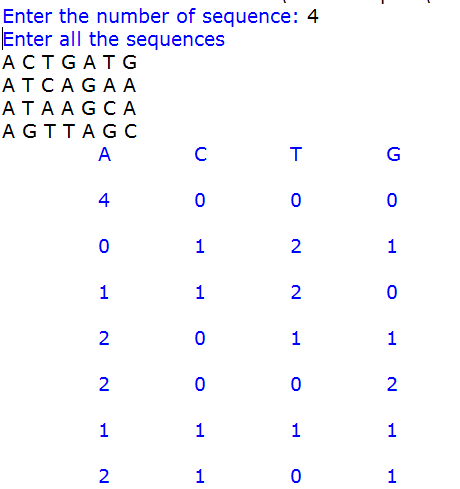
seq\_list=[]

for \_ in range(no\_of\_seq):

seq\_list.append(list(map(str, input("").split())))

solve\_fingerprint(seq\_list,len(seq\_list[0]))

**Output**



**Practical 8- Find Regular Expression of given sequences.**

**Source code:**

def gen\_reg\_exp(seq\_list, no\_of\_col):

final\_list=[]

for colnum in range(no\_of\_col):

collist=[]

for colseq in seq\_list:

collist.append(colseq[colnum])

if len(set(collist))==len(collist):

#print(final\_list)

final\_list.append('x')

else:

if len(set(collist))==1:

final\_list.append(collist[0])

else:

final\_list.append(''.join(set(collist)))

display\_output(final\_list)

def display\_output(final\_list):

print(\*final\_list, sep='-')

no\_of\_seq=int(input("Enter the number of sequence: "))

print("Enter all the sequences")

seq\_list=[]

for \_ in range(no\_of\_seq):

seq\_list.append(list(map(str, input("").split())))

gen\_reg\_exp(seq\_list, len(seq\_list[0]))

**Output:**

